SD1102: Hand-2-Speech

Final Report

Mohammed Albalawi, Davis Beattie, Stephen Farnsworth

Advisor: Dr. Ababei

Background

The Hand-2-Speech device is designed to provide a new way for deaf people to communicate using sign language. This technology was first designed by Colgate University. The device takes sign language symbols and coverts them into text and speech. This device could be a very useful learning tool for those who want to learn sign language. The Hand-2-Speech project was proposed by Dr. Cristinel Ababei at North Dakota State University.

Problem

Very few people understand sign language. Often, deaf people are forced to use written notes or ambiguous gestures to communicate with people who do not know the language. The Hand-2-Speech device was proposed to bridge this gap. Those who know sign language could use its capabilities to convert their signs into typed text or verbal speech others could hear. Also, those who do not know sign language could use the device to learn the language. The Hand-2-Speech helps solve this communication gap.

Requirements

- The device must use sensors to detect the motions of the hand
- The device must be wireless, sending from a module on a glove to a receiver
- This receiver must interface with a PC using USB
- The device must recognize words and phrases and "say" them using speech
    - The speech element may come from a previous program
- The glove must be battery powered and the module must be compact enough to fit on a hand.
- The hardware must all be custom designed

System Block Diagram

The device is divided into three parts: The glove, the USB device, and the text to speech program. The glove interprets data from various sensors and determines a signed letter. The glove then sends this letter to the USB Device. The USB device sends this letter to the USB port of a PC as a typed letter, like keyboard. The text to speech program finally converts the text into speech. The flow of the system is shown in the block diagram in Figure 1.

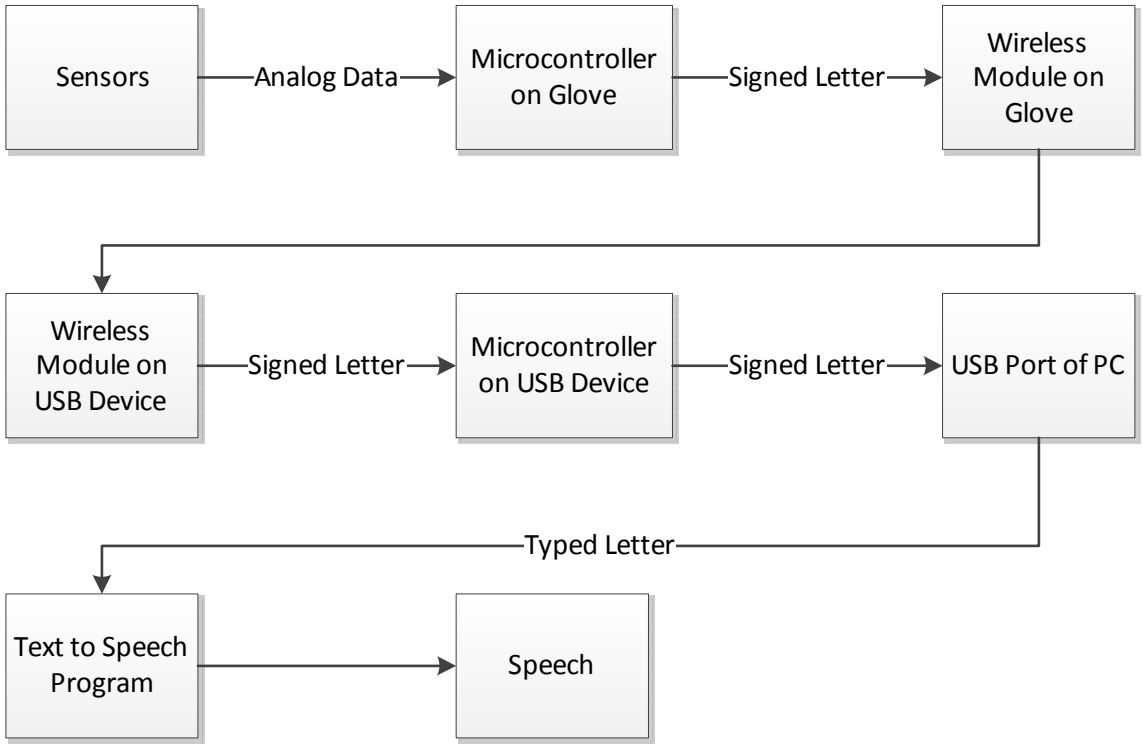

Figure 1: Block Diagram of Entire System

Hardware Design and Schematics

Parts List

Glove Expense			
Quantity	Item	Unit Price	Amount
1	GLOVE	$19.00	$19.00
2	PIC18F4550	$4.00	$8.00
5	FLEX RESISTORS	$12.50	$62.50
1	OP AMP	$1.50	$1.50
1	VOLTAGE REGULATOR 5 VOLT	$2.75	$2.75
1	VOLTAGE REGULATOR 3 VOLT	$2.75	$2.75
1	MULTIPLEXER	$3.50	$3.50
2	PROFESSIONAL PCB	$53.00	$106.00
XX	MISC. (RESISTORS, CAPS, CRYSTALS)	$19.00	$19.00
		Total =	$225.00

Figure 2: Glove Parts

The first page of the data sheets of the main electrical components is located in the appendix.

Glove

Signed letters are determined using flex sensors on each finger and an accelerometer. The flex sensors change resistance based on the amount of bend in the sensor. To measure the bend, the flex sensors are attached to a voltage divider to convert resistance into voltage. The voltage is sent to an analog to digital converter on the microcontroller. This requires some amplification and a multiplexor, which are shown in Figure 2 below. The accelerometer outputs a straight voltage for each axis (3) and needs no extra steps to send the voltage to the analog to digital converter. Together, these sensors can accurately detect and output the position of the fingers and hand using voltages.

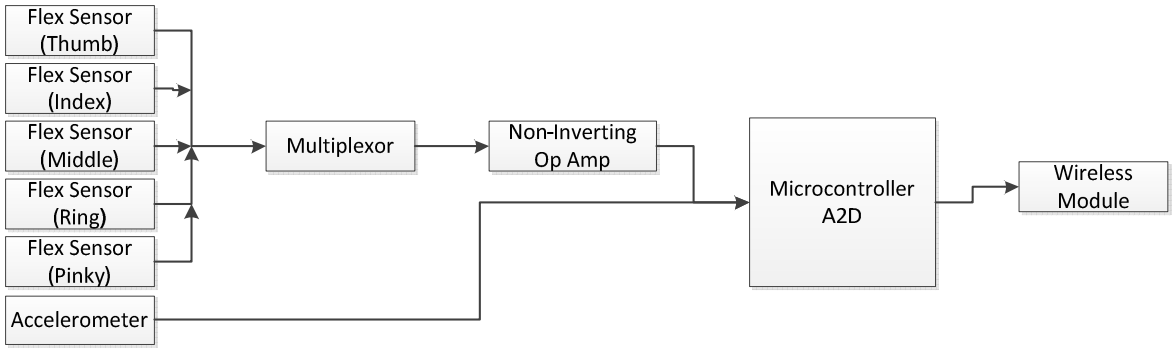

Figure 3: Block Diagram of Glove Sensors

Flex sensors are rated for resistances of around 10kΩ to around 20kΩ to 100kΩ depending on the sensor. To normalize this, we chose to place a resistor in parallel with the higher resistance sensors to lower their overall resistance. A target of 25kΩ maximum was chosen as the lower resistance sensors maxed out around that value. Figure 3 below shows the resistance values of the sensors, the chosen parallel resistors, and the final equivalent resistances of the sensors.

	Min	Max	Parallel Resistor	Final Min	Final Max
Thumb	9.7k	38k	68k	8.5k	24.4k
Index	12.2k	20k	None	12.2k	20k
Middle	10.6k	56k	47k	8.7k	25.6k
Ring	8.8k	19k	None	8.8k	19k
Pinky	15.3k	82k	33k	10.4k	23.5k
			Overall	8.5k	25.6k

Figure 4: Flex sensor resistance values

A voltage divider and non-inverting op amp circuit are used to obtain a reading and amplify it to the maximum range of the analog to digital converter, 0 to 5 volts. The op amp uses an offset on the

negative terminal to set the minimum input value to 0V. Because our system also has a 3.3 volt voltage regulator, we chose 3.3V to be this offset and designed our voltage divider and op amp based on it.

The voltage divider resistance was calculated using the following:

$$3.3V = 5V \times \frac{8k}{R + 8k}$$

with 8k being the minimum voltage of all the flex sensor from Figure 3. R = 4.1kΩ as a result. However, there is no resistor made at that value, so 3.9kΩ was chosen instead. The amplifier gain was calculated as follows:

$$Gain = \frac{5}{4.35 - 3.36} = 5.05$$

with 4.35 being the maximum possible voltage output from the voltage divider using a 3.9kΩ resistor and the largest value from the flex sensors in Figure 3, and 3.36 being the minimum. This results in a gain of around 5. We chose resistance values of 750kΩ and 150kΩ to achieve this gain, and to reduce the current draw of the system. The entire circuit is shown in Figure 4 below.

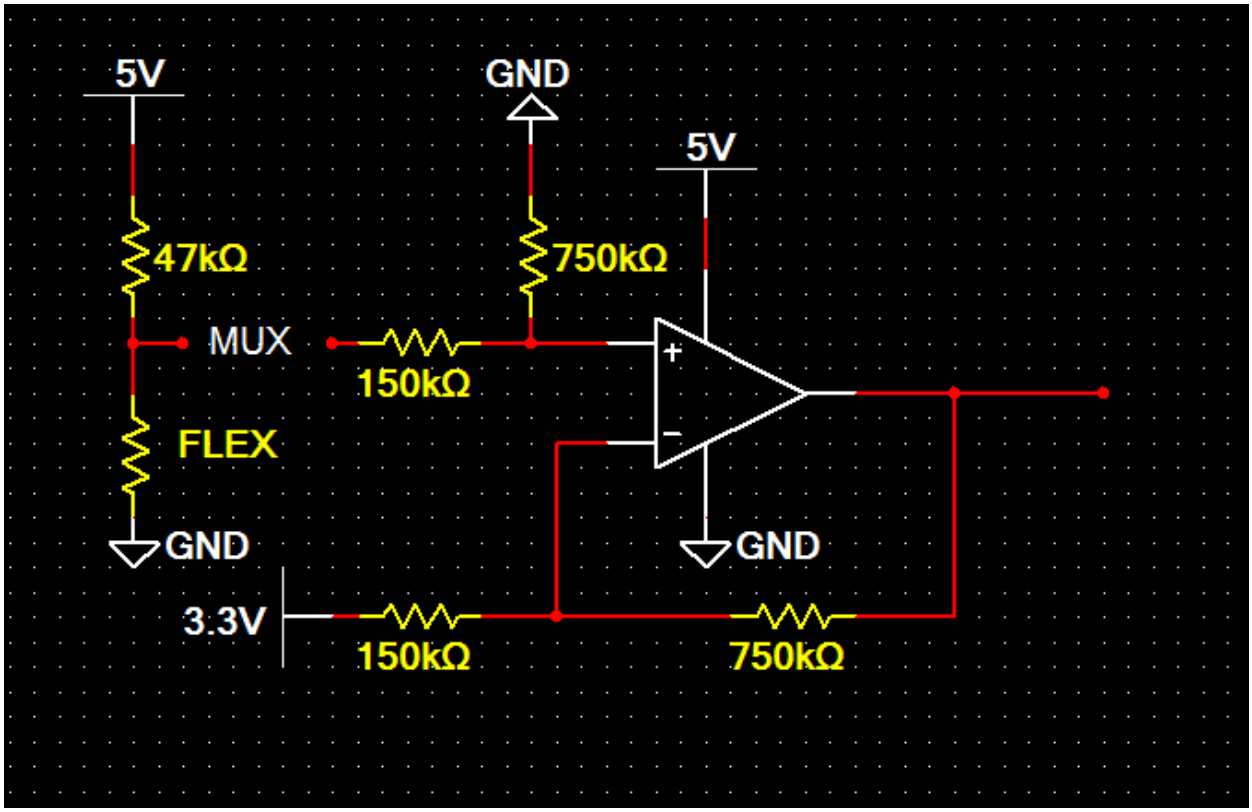

Figure 5: Flex sensor and op amp circuit

The multiplexor uses the RB1, RB2, and RB3 digital outputs from the microprocessor to switch between the 5 flex sensors. All the flex sensors will pass through the same op amp, so this saves footprint space.

The accelerometer is rated to output 0V-3.3V depending on the amount of acceleration measured in a particular axis. Our system only requires us to know which direction the hand is facing, so gravity is our

main measure. This results in a range of 1.1V to 1.8V. Because these values are consistent and steady, we chose not to amplify these values and save the footprint space. The accelerometer circuit is shown in Figure 5.

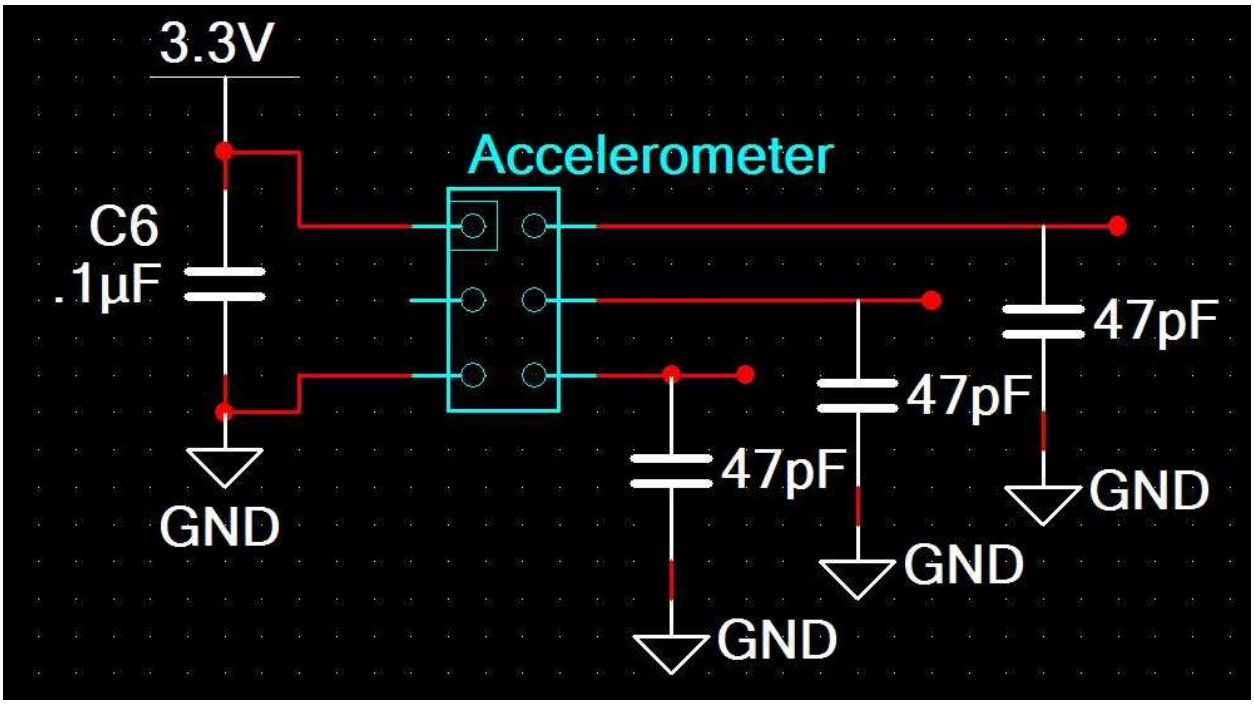

Figure 6: Accelerometer

Power

The entire system is powered using a 9V battery. The battery powers a 5V regulator and a 3.3V regulator. The 3.3V regulator powers the accelerometer and the op amp offset and the 5V regulator powers everything else. According to the average ratings on the data sheets, each regulator has sufficient power to handle everything. The 9V battery are rated around 500mA/hr, so the system should last over 100 hours on a single battery. The ratings are shown in Figure 6 and the circuits shown in Figures 7 and 8.

System Current Draw			
	Input		Output
TPS7150Y 5V Regulator	285µA		500mA
PIC18F4550	25mA		
SN74HC481 MUX	2µA		
MCP602 Op Amp	230µA		
Xbee RF Module	45mA		
5V to Flex Sensor (5)	20µA		
LP2950 3.3V Regulator	75µA		100mA
ADXL335 Accelerometer	350µA		
3.3V to Op Amp	4µA		

Figure 7: System Current Draw

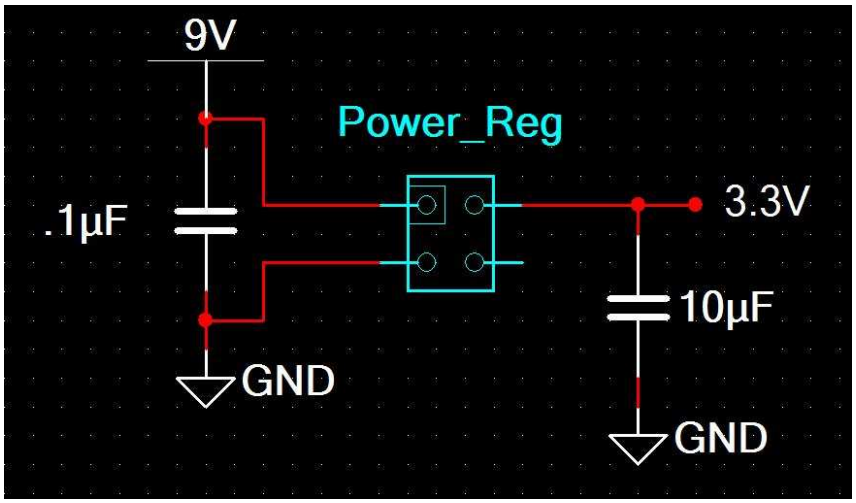

Figure 8: 3.3V Regulator Circuit

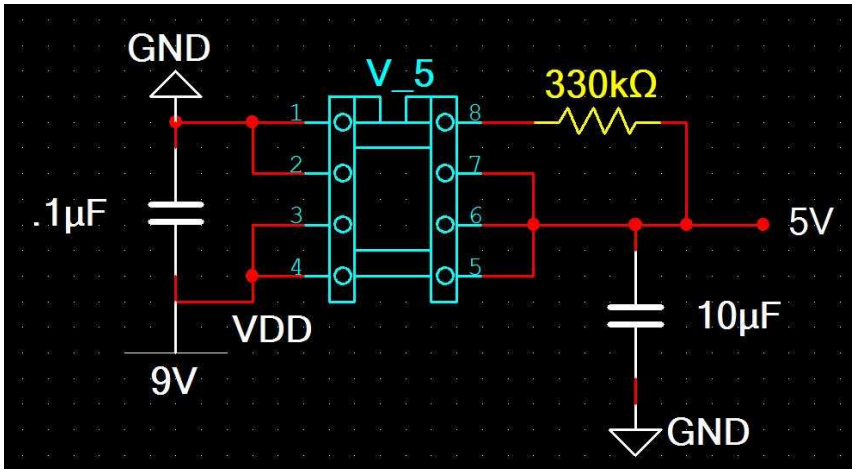

Figure 9: 5V Regulator Circuit

The microcontroller is the centerpiece of the glove circuit. It uses a 20MHz crystal for timing and is powered with 5V. Four analog input channels are used to read in data: one for the multiplexor and three for each of the accelerometer outputs. The SCI port (pins 25 and 26) connects to the Xbee breakout board, which houses the Xbee wireless module. This pins are set so the transmit pin of one device connects to the receive pin of the other. Data can be sent in standard SCI form and the Xbee sends it out like a wire. RB1, RB2, and RB3 connect to the multiplexor to select which flex sensor is to be output. The circuit is shown in Figure 9.

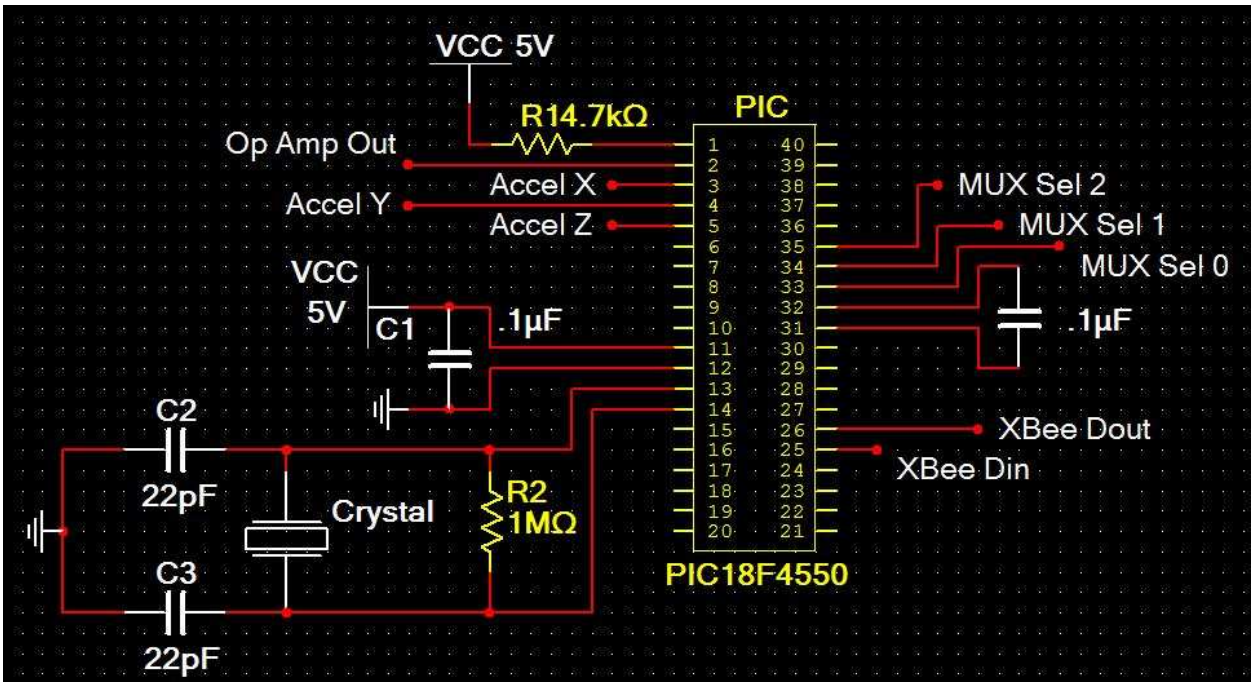

Figure 10: Glove microcontroller circuit

USB Device

The USB device receives data wirelessly from the glove and interfaces with a PC to type the signed letter to the screen. The PIC18F4550 has built in USB circuits to simply interface. The only hardware requirement is a 470nF capacitor from pin 18 of the PIC to ground. This microcontroller also uses a 20MHz crystal and 5V power. The Xbee is also set to pins 25 and 26 just like the glove microcontroller. The USB is connected in a similar fashion, just with pins 23 and 24. The USB device only needs to interact with the Xbee and the USB, so it is a much smaller circuit than the glove. The circuit design is shown in Figure 10.

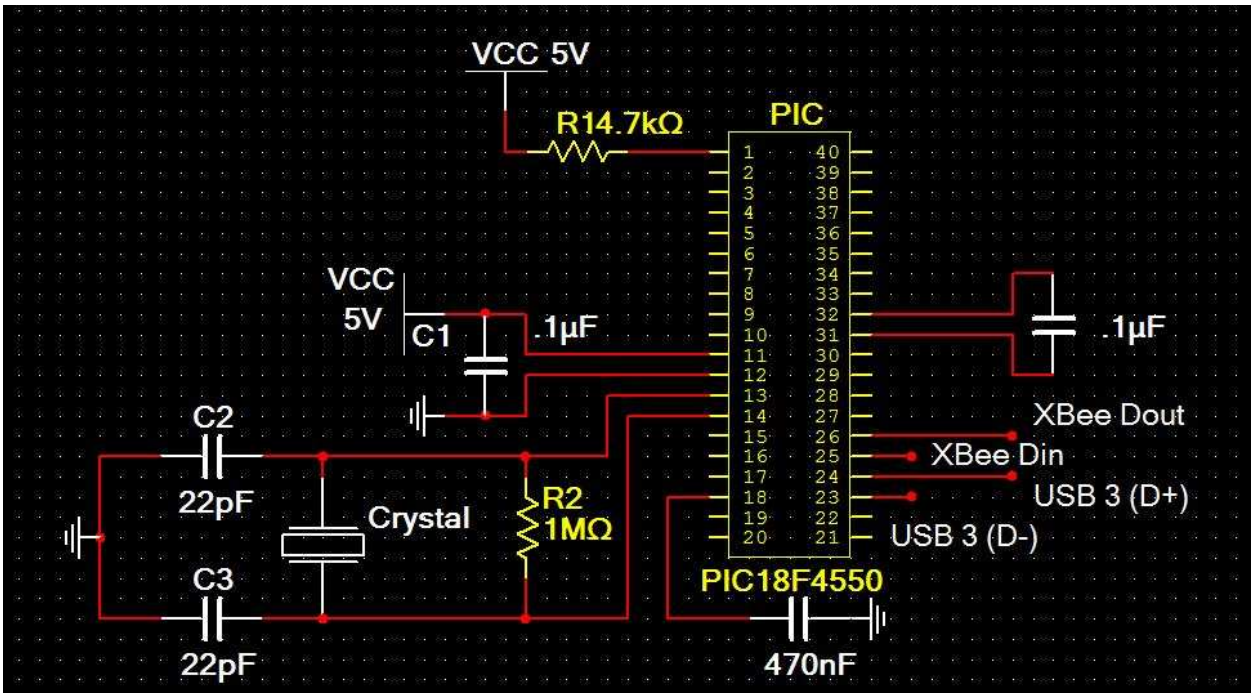

Figure 11: USB device circuits

PCB Design

The printed circuit board (PCB) design was first done via the schematic done in Multisim 11.0. Once the schematics were checked over multiple times for accuracy they were transferred to Ultiboard 11.0. In Ultiboard the components got laid out in a configuration that utilized the space properly while trying to minimize the number of vias and traces. The schematics and PCB layouts are in Figures 11 through 14 both the glove circuit and USB circuit, respectively.

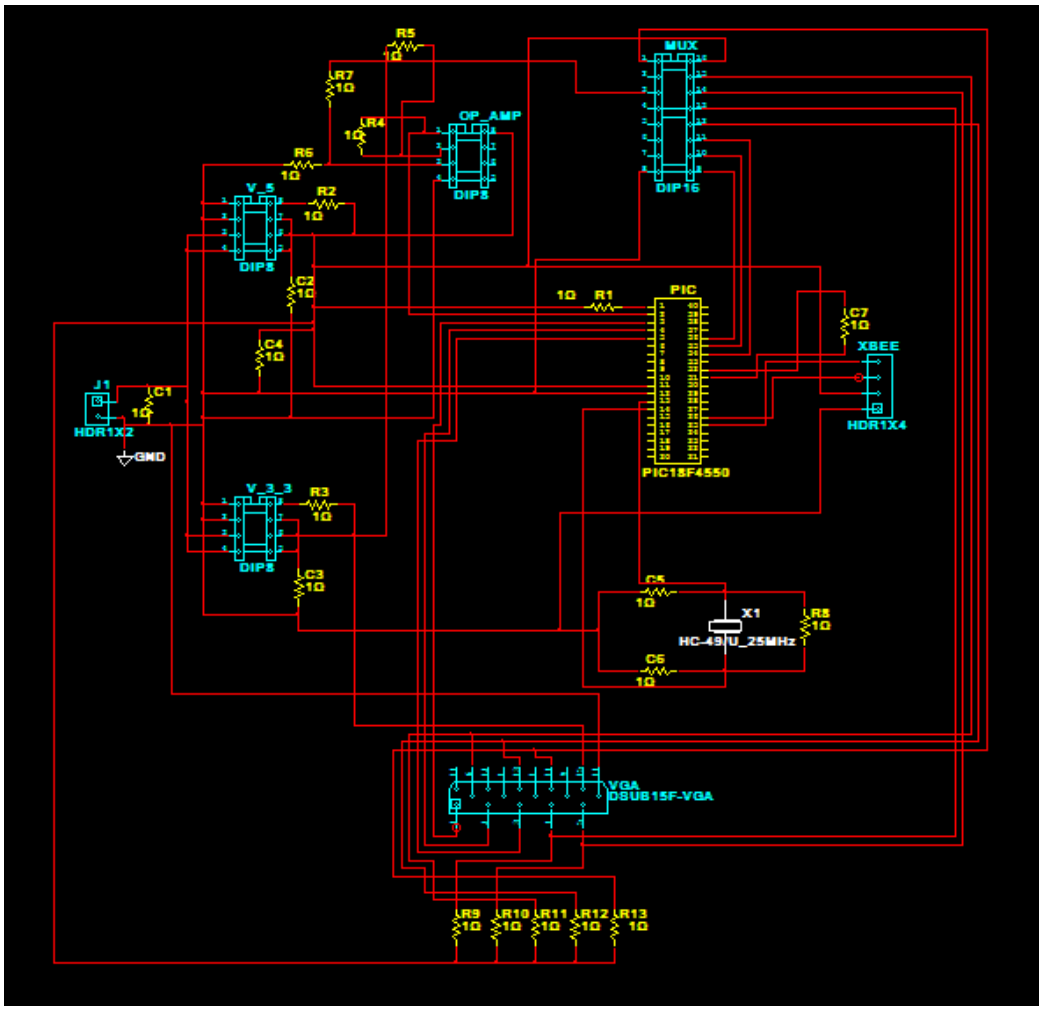

Figure 12: Glove schematic

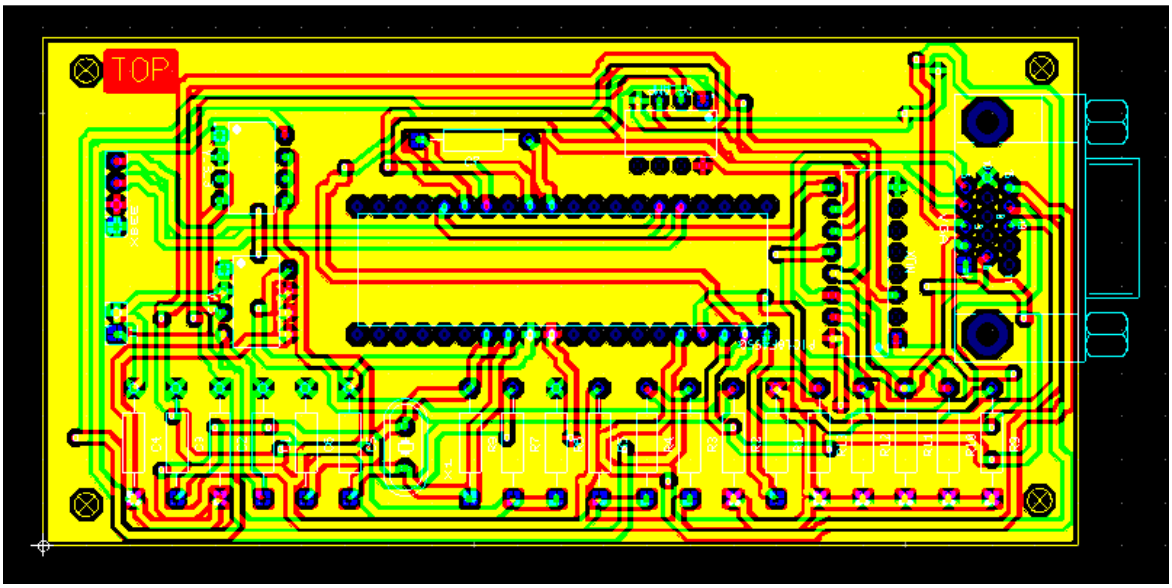

Figure 13: Glove PCB

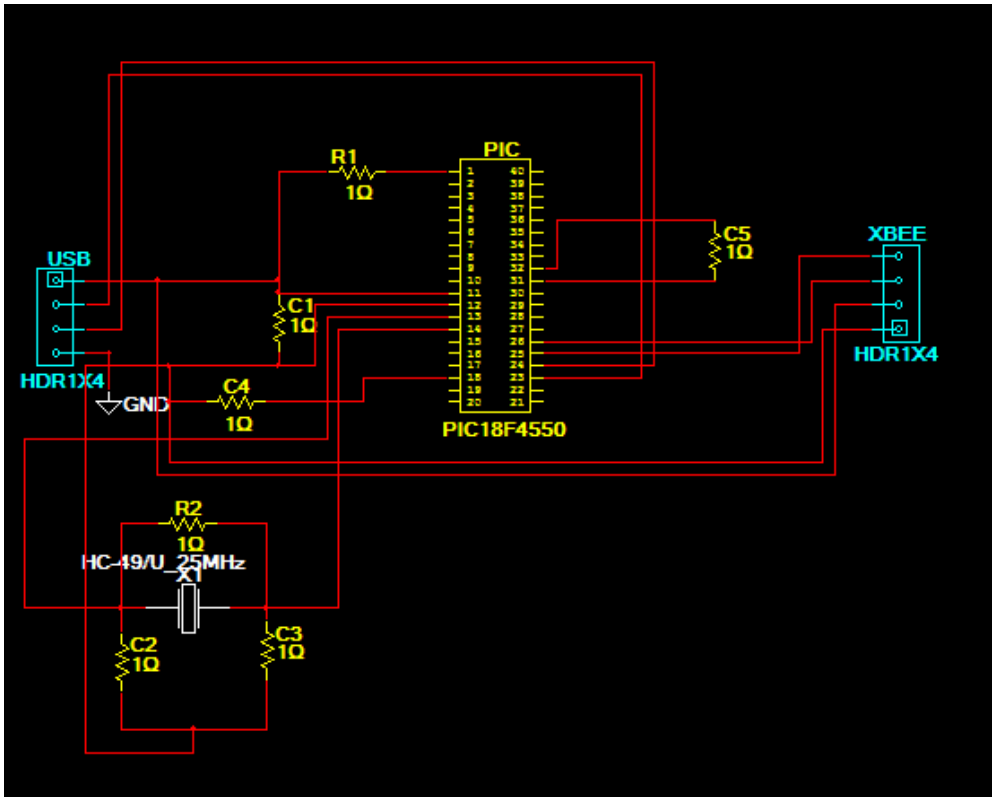

Figure 14: USB schematic

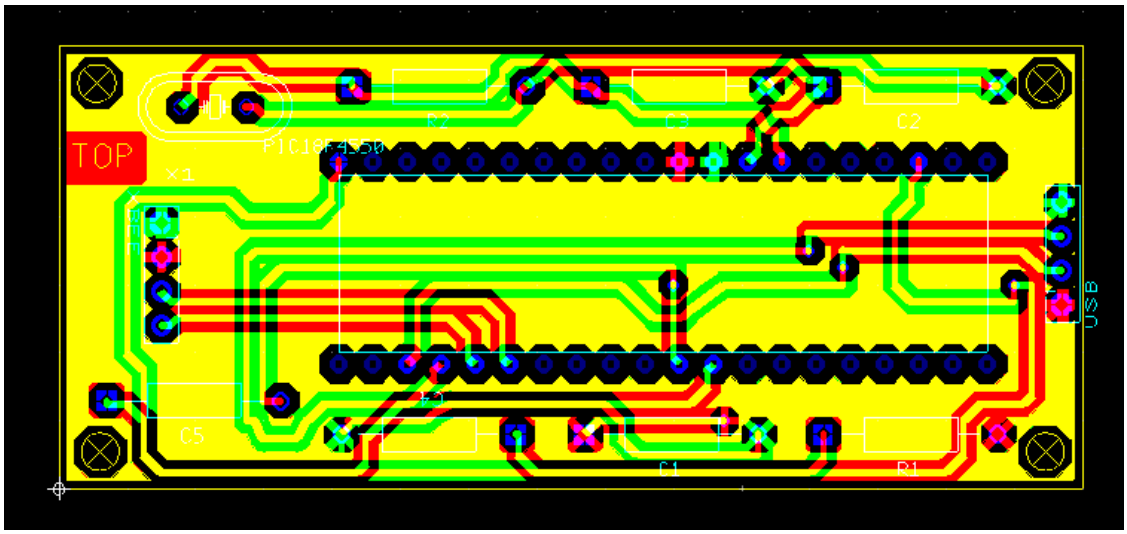

Figure 15: USB PCB

Final Assembly

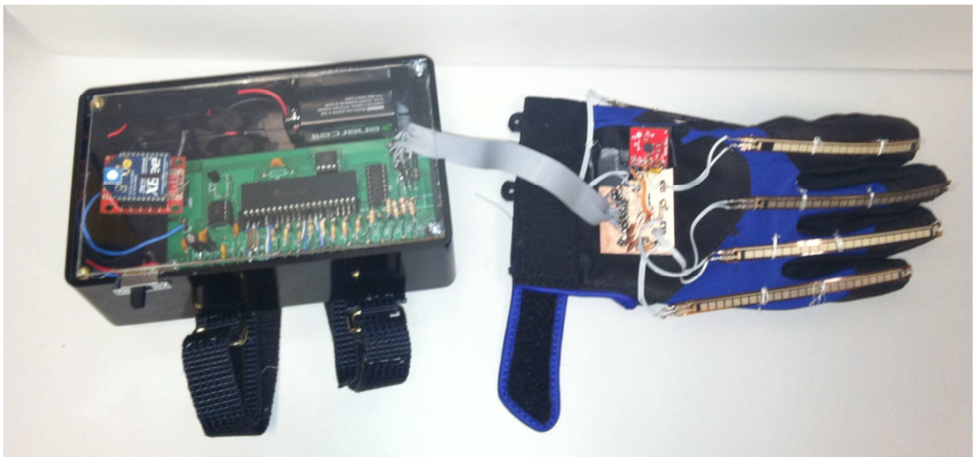

Figure 16: Glove and glove enclosure

Each flex sensor was sewn at the tip of the finger and held in place by my thread at the joints of the fingers to assure their ability to slide smoothly, as shown in figure 15. Each flex sensor was connected to the interface board with ribbon wire, very thin wire, to not bind up the resistors. The interface board housed the accelerometer and connected all the sensors. Ribbon wire connected the interface board to the glove PCB for the same reason as before, to assure flexibility. The glove PCB enclosure has two straps to enable the user to easily attach the housing unto their wrist.

Figure 17: USB enclosure

In figure 16, the USB enclosure houses the receiving wireless signal that is then connected to the computer. Once the USB is connected to the computer so signing can appear in any text document. This will be discussed in greater detail in the following sections.

Software Design

The software is divided between the two modules: the glove and the USB device. The software recognizes which symbol is being signed, sending that symbol wirelessly to the PC, and typing that symbol on screen. (Note: Complete code can be found in the Appendix)

Glove

The glove code can be divided into three sections: Analog to digital conversion, letter determination, and wireless transmission. A flow chart of the code is shown below.

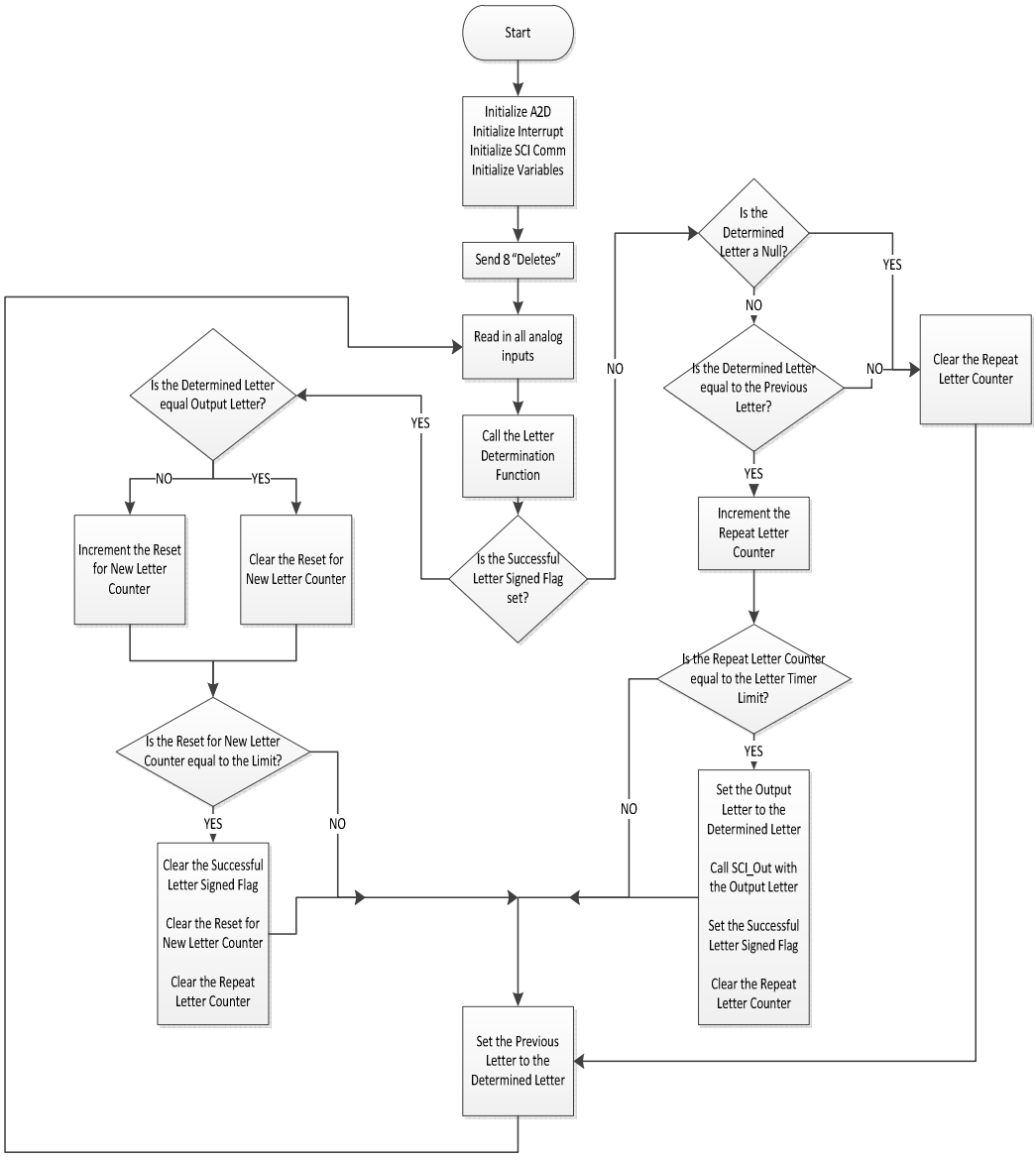

Figure 18: Glove Flow Chart

Analog to Digital Conversion. The glove uses four (4) channels for analog to digital conversion. Channel 0 is connected to the multiplexor and handles all the flex sensor inputs. The code sets the multiplexor to

the thumb input by means of Port B, and then reads the analog input. This repeats for all five (5) fingers. Then, the code reads in Channel 1, the X-axis of the accelerometer, Channel 2, the Y-axis, and Channel 3, the Z-axis. These readings are stored in variables that won't be updated until the loop repeats itself.

Letter Determination. Once all the analog inputs have been read, the relevant inputs are passed into the Letter Determination function. This function outputs the enumeration of a letter or symbol if a letter is determined. 0 is defined as a null output, while 1-26 are reserved for the letters. Delete and the Bison symbol are defined as 27 and 28. The idea behind the Letter Determination function is that, if all six (6) sensors (five (5) flex sensors plus the X-axis) fall within the ranges of a certain symbol, that symbol is output. These ranges were predetermined by testing and reading the values of every sensor while a symbol was made. Data was taken over time, and the average value was determined to be that symbol. All the values are defined in a header file, including the upper and lower limits of the range.

The function is a series of else-if statements testing each sensor value. For example, say the predetermined thumb value for the letter A was 400. The range is determined to be plus or minus 70 from that, so the range is from 330 to 470. A user signs the letter A and the analog to digital conversion is read as 440, so this is read as true. This occurs for all six (6) sensors and if all six (6) match, that letter is output. If any sensor does not fall into the range, that letter is not output.

Because of the else-if structure, some letters take priority over others, as once a letter has been determined; the rest of the function is skipped. We chose to use Scrabble points to determine the order, with common letters first and rare letters last. The delete symbol was placed first and the Bison symbol last.

Wireless Transmission. After a letter has been determined or not, there is a series of logic to determine if the letter is to be output wirelessly. This is divided into two (2) sections, if a letter hasn't just been wirelessly output (right side of the flow chart above), or if a letter has just been output (left side of the flow chart).

If a letter hasn't just been output, then the code is looking for a symbol to be repeated a certain number of times without any interruptions. This way, a symbol must be held before it being output. If the same letter is determined two loops in a row, a counter is incremented. If at any time a different letter or a null is recognized, that counter is reset. If the counter reaches the threshold, the code will output the appropriate letter. This is done by sending the letter as an ASCII character (i.e. 'A') over the serial port to the Xbee. The delete is set to character '1,' while the Bison symbol is set to '2.' The serial port communication is initialized to 9600 baud. After the letter is output, a flag indicating a letter has just been output is set.

While this flag is set, the code described in the above paragraph is ignored. Instead, a set of code essentially the opposite of it is used. The analog to digital conversion and letter determination sections are run as usual, but instead of checking to see if the letters are the same as the previous loop, this section checks if they are different. A counter is incremented if the newly determined letter is different

from the just wirelessly transmitted letter. This counter is reset if the letter is the same. Once this counter reaches the limit, the flag indicating a letter has just been output is cleared. Now, the code is once again ready to output another letter.

Using this design, the code allows for the user to hold their hand in a symbol and not have the program output that symbol repeatedly. The user must remove their hand from that symbol in order for a new one to be signed, including that same symbol. This is consistent with sign language practices. This code is compiled using Hi-Tech Compiler in MP Lab

USB Device

The USB software is based on the Microchip HID Keyboard Demo provided with MP Lab. The code is modified to receive wireless data and use that data to output to the keyboard. The flow chart of the code is shown below.

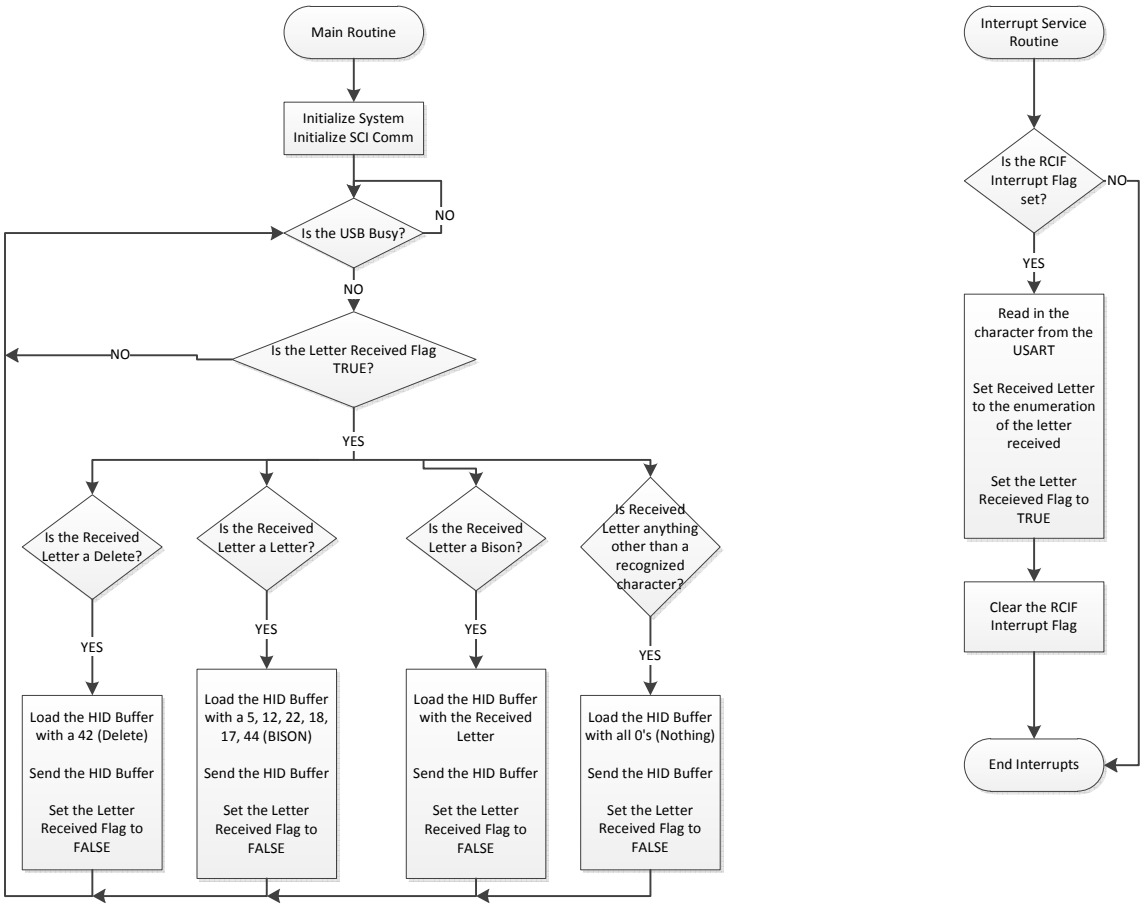

Figure 19: USB Device Flow Chart

The wireless is received using an interrupt. Anytime a message has been received by the serial port from the Xbee, the RCIF interrupt is set and executes. This interrupt simply recognizes the received character and assigns a global output variable to that letter. For example, if 'a' is received, the output

variable is set to 4. The keyboard demo presets A-Z as 4-29. Delete is set to 3 and Bison is set to 2. This output variable is passed to the main code.

The main code simply waits for a flag to be set within the interrupt routine. Thus, when a letter is received, the main code decides what to type to the screen. This is done by loading an HID buffer. This buffer can send 6 characters at once. If a letter is received, that letter is output plus a space. If a delete is received, then a delete is sent. Finally, if a Bison symbol is received, "bison" plus a space is output.

This code is compiled using C18 compiler and MP Lab.

Text to Speech

Because our requirements allow for previously made software for the speech aspect, we chose to allow the device to type into any program that accepts text. We use Google Translate in English, which will say the letters that are typed into the interface. This method sacrifices instant feedback, but they can be undesirable with the sensitivity and difficulty of sign language to a new user. We also retain more functionality being able to use any program or interface.

Technician's Troubleshooting

The following section will detail normal operating, typical problems and troubleshooting, and repair of the device. These are problems we came across during testing of the device.

Normal Operating

The USB device should be plugged into the computer first. Because the device is HID, there are no drivers to install and the device will work on any PC. When the glove is turned on, it first sends out a "1 2 3" from the bootloader. This causes the USB device to output "Bison" and a delete. The glove initialization sends deletes to remove these letters, so this is the first indication that the device is working.

At this point, the glove can begin recognizing symbols. A simple letter to ensure this is functioning is the letter "A," a simple fist with the thumb on the outside of the hand. The device should type the letter once and stop until the hand is removed from the position. The delete, which is done by tilting the hand inward (counter-clockwise), is also a simple symbol to test functionality.

To turn off, simply turn off the switch and removed the USB device from the PC.

Typical Problems and Troubleshooting

Flex Sensor Failure. One side effect of our letter determination design is that if a single sensor fails, no letters can be determined. The way to discover if this is the case is to sign a simple letter, such as "A." If

after multiple attempts no letter is being output, the most common problem is that a sensor has failed. Typically this occurs at the connecting point between the flex sensors and wires. Toggling power on the device will still allow "bison" to write to screen and be deleted in this case.

PIC Failure. If the PIC fails, the initializing "bison" will not be typed onto the screen. This is the most common sign of a PIC failure. Also, the power light on the Xbee board will be lit up, indicating that power is present. Typically, the PIC must be reprogrammed or replaced if a failure has occurred.

Power Failure. If no "bison" is typed to the screen on start-up and the light on the Xbee board is not on, this indicates a power failure. The battery may need to be replaced or a voltage regulator may be malfunctioning.

Repair

In the case of flex sensor failure, the connecting wires may need to be soldered, either to the sensor itself or the interface board itself. If this does not solve the problem, the flex sensor itself may need to be replaced. If a PIC failure has occurred, it must be replaced. Simply remove the broken from the socket, and replace with the new, programmed one. Power failure simply requires a battery replacement.

Project Comments

Project Issues

There were many issues in the development of the Hand-2-Speech. The first was understanding American sign language. We had to decide to use the alphabet, as it only requires the use of one hand and relative hand position is not important to recognize it. We also had to understand component compatibilities. We originally designed the code on a different PIC, but the 4550 is the only one that is USB compatible. There were some issues with our Xbees as well. We programmed them to only talk with each other to avoid interference. The last major issue was letter calibration. Any time the project was moved, whether from breadboard, to prototype glove, to final glove and PCBs, the resistance values of the flex sensors changed. This required complete recalibration.

Design Flaws and Future Improvements

A first design flaw is with our PCB design. The 40-pin socket had to be created in Ultiboard for our project. However, our pin numbers were flipped on the actual printed board, so we had to solder jumpers around the board to get the correct result.

Our accelerometer wore out, so only the X-axis output worked by the time the final product was put together. The project could be improved by having a fully functioning accelerometer and utilizing all the outputs for better accuracy in recognizing symbols.

The flex sensors on our final glove have wildly different resistance values. Our code would be more accurate if we had flex sensors with very similar resistances. Unfortunately, this requires trial and error, and we can't guarantee the values in the flex sensors we order.

Overall, the entire forearm box is too large and cumbersome. The PIC's could have been twenty pin, which would have made the entire system smaller. Also, the enclosures could have been shorter. Changing these would make it less tiresome to use the Hand-2-Speech.

Our system has some issues recognizing some letters, as they are similar to each other. Specifically, R, U, and V all have straight index and middle fingers with the other fingers in a fist. Our design of using accelerometers and flex sensors does not create a large difference between these. Adding touch sensors to the fingers would greatly improve these three letters' recognition, as well as many of the others.

Finally, our final system samples at too quickly of a rate. Currently, it is uncontrolled. Our system can keep up, but too well. For example, when trying to sign an O, the hand has to pass through the letter C. Because our system samples too quickly, the letter C is recognized when it isn't intended to be. A solution to this is controlling how many samples per second the system reads, and then resetting our threshold counters to a reasonable value.

Lessons Learned

- Planning and communication are key.
- Setting deadlines ensures continuous project progress.
- Dividing work is important.
- Often, the solution is simpler than it appears.

Projects That Could Be Derived From Hand-2-Speech

- Two glove system that recognizes the full American Sign Language
- Hand operated gaming control
- Gesture mouse for PC
- Remote control for devices/robots
- General device controller for disabled

Design Tips

- Set strict deadlines
- Divide the work
- Do ample planning before ordering any parts or building
- Play to your group members' strengths